

Optimal Software Patching Plan for PMUs

Teklemariam Tsegay Tesfay, *Student Member, IEEE*, Jean-Yves Le Boudec, *Fellow, IEEE*, and Ola Svensson

Abstract—Phasor measurement units (PMUs) deployed to monitor the state of an electrical grid need to be patched from time to time to prevent attacks that exploit vulnerabilities in the software. Applying some of these patches requires a PMU reboot, which takes the PMU offline for some time. If the PMU placement provides enough redundancy, it is possible to patch a set of PMUs at a time while maintaining full system observability. The challenge is then to find a patching plan that guarantees that the patch is rolled out to all PMUs in the smallest number of rounds possible while full system observability is maintained at all times. We show that this problem can be formulated as a sensor patching problem, which we demonstrate to be NP-complete. However, if the grid forms a tree, we show that the minimum number of rounds is two and we provide a polynomial-time algorithm that finds an optimal patching plan. For the non-tree case, we formulate the problem as a binary integer linear programming problem (BILP) and solve it using an ILP-solver. We also propose a heuristic algorithm to find an approximate solution to the patching problem for grids that are too large to be solved by an ILP-solver. Through simulation, we compare the performance of the ILP-solver and the heuristic algorithm over different bus systems.

Index Terms—Software patching, Observability, Phasor Measurement Unit, Vulnerability, NP-complete, Heuristic.

I. INTRODUCTION

The information and communications technology (ICT) infrastructure in a smart grid network consists of a large number of heterogeneous field devices and servers running a variety of software systems. Utilities deploy state-of-the-art cybersecurity solutions to fend off attacks against the ICT infrastructure. However, no matter how strong the deployed security solutions are, the fact remains that there is no fool proof solution that provides absolute security against all possible attack vectors. There will always be unknown vulnerabilities in the software or hardware that an attacker will discover and exploit through time to compromise one or more of the devices.

Therefore, in addition to deploying state-of-the-art security solutions and taking reactive measures whenever there is a cyber attack, it is important for utilities to take pro-active measures, such as putting a software patch management in place. Deploying an efficient patch management process for industrial control systems (ICSs) has been addressed in [1]–[3]. A software patch management system guarantees that patches are applied to all devices running the vulnerable software. It is important that software patches that fix vulnerabilities are rolled out uniformly to all devices as soon as they are available. That is because if a patch is not applied on time and the vulnerability is of public knowledge to an attacker, the

attacker will compromise one of the devices by exploiting the vulnerability. Once an attacker gets access to one such device, she can maintain access to the device by privilege escalation even after the patch is applied later on. By maintaining access to the device, the attacker can exploit the trust relationship the device has with other communicating partners in order to launch further attacks and compromise more devices in the network.

In light of the need to roll out software patches fast to all devices, we study the problem of software patching for phasor measurement units (PMUs) in smart grids. PMUs measure time-synchronized, high-resolution phasor data from several locations of the grid and stream this data to a central location called phasor data concentrators (PDC). The PDC time-aligns the measurements from the different PMUs and feeds the time-aligned synchrophasor data to a real-time state estimator.

Since a PMU placed in a particular bus measures the bus' voltage phasors as well as the current phasors of all the branches incident to the bus, Kirchhoffs laws make it possible for the PMU to indirectly measure the voltage phasors of all incident buses. Therefore, the total number of PMUs required for full system observability is less than the total number of buses in the network. The required number of PMUs is even smaller when we take the presence of zero-injection buses and conventional measurement devices into consideration. Finding an optimal PMU placement that minimizes the number of PMUs that provide full system observability is a widely studied problem. Research done to address this problem can be broadly categorized into two groups [4]: (1) deterministic approaches that formulate the problem as an ILP problem satisfying some constraints [4]–[11] (2) meta-heuristic algorithms [12]–[16].

While deciding on an optimal placement of PMUs to a grid, a utility normally adds a contingency constraint that ensures that the placement provides full observability even when any one of the PMUs fails or is offline for maintenance purposes. Adding more PMUs than the minimum number required for observability also increases measurement redundancy, which improves a state estimator's accuracy as well as its ability to detect bad data [17]. A PMU placement that provides enough measurement redundancy also enables a utility to roll out a software patch to all PMUs by patching a subset of the PMUs at a time while maintaining system observability at all times. In a large-scale power system that deploys a large number of PMUs, applying the patch to one or only a few PMUs at a time is infeasible. As stated above, if the patch roll-out takes an extended amount of time, an attacker may exploit the vulnerability to quickly launch an attack on the not yet patched PMUs. The exploited vulnerability could be similar to OpenSSL's Heartbleed bug [18], which can be used by an attacker to steal confidential information like passwords

Teklemariam Tsegay Tesfay, Jean-Yves Le Boudec and Ola Svensson are with the École Polytechnique Fédérale de Lausanne (EPFL), CH-1015, Lausanne, Switzerland.

This work was supported by the Swiss National Research Program Nano-Tera.ch.

and secret keys. The effect of such vulnerability is long-lived even after the vulnerability is fixed. In addition to the security concerns discussed above, such a slow patch roll-out is ineffective because it demands a great deal of time for the personnel responsible for the task. It is also inefficient in terms of network bandwidth utilization because streaming a large software patch in several rounds may have significant effect on the delay-sensitive communication involving phasor measurement data. Moreover, not having all PMUs patched and getting them back to operation as fast as possible results in a reduced measurement redundancy, which may have a negative effect on the quality of a state estimator's output.

The main challenge we address in this chapter is, therefore, a patching plan that minimizes the number of rounds required to patch all the PMUs without losing full observability of the grid during the entire time. Stated otherwise, our goal is to find a partitioning of the set of deployed PMUs into as few subsets as possible such that all the PMUs in one subset can be patched at a time while all the PMUs in the other subsets along with any available conventional measurements provide full observability of the system.

The main contributions of this paper are:

- We formulate the PMU patching problem as a sensor patching problem and show that the problem of finding an optimal sensor patching plan is NP-complete.
- For the case when a power grid has a radial structure (is a tree), we show the minimum number of rounds required to patch all deployed PMUs is equal to two. We also provide a polynomial-time algorithm that finds the optimal patching plan.
- For mesh grids (non-radial structured grids), we formulate the sensor patching problem as a binary integer linear programming (BILP) problem and use a branch-and-bound based ILP solver to compute a patching plan for different bus systems. For grids that are too large to be solved by the ILP-solver, we propose a greedy heuristic algorithm to compute an approximate solution. Moreover, we have proved that finding an optimal solution to the problem is equivalent to maximizing a submodular set function.

Although we study the problem as a planning problem for the offline time of PMUs caused by software patching, it can be generalized to any scheduled maintenance work that affects all PMUs and requires a PMU to go offline for some time.

The rest of this paper is organized as follows. In Section II we state assumptions, introduce the system model and define the PMU patching problem. In Section III, we formally define the PMU patching problem as an instance of a sensor patching problem using set theoretic approach. We also show it is NP-complete. The BILP formulation of the problem using the asymmetric representatives method is also introduced in this section. In Section IV we introduce a polynomial-time algorithm that finds an optimal two-round patching plan on a tree and prove its correctness. Section V discusses the heuristic algorithm for the general case networks. Results from the heuristic approach and the ILP solver are presented and compared in Section VI. Section VII provides concluding remarks.

II. PMU PATCHING PROBLEM

In this section, we briefly describe our assumptions on state estimation and system observability. We also introduce the system model and define the PMU patching problem.

A. State Estimation and Assumptions

The static estimation of a power system state is defined as determining the voltage magnitudes and phases at all the system buses through analysis of measurements collected from different locations of the grid [19]. The state estimator uses the set of measurements along with the power system model as an input to compute the most likely state of the grid at a given time. The set of measurements may include phasor data from phasor measurement units (PMUs) and conventional measurements from P-Q measurement devices. A phasor measurement unit (PMU) is a device that directly measures nodal voltage magnitudes and phase angles and branch current magnitudes and phase angles. We assume that a PMU placed at a bus has enough number of channels to measure a bus' voltage phasor as well as the current phasors of all lines incident to the bus. A P-Q measurement device may be an injection measurement or a flow measurement device. An injection measurement device measures the real and reactive nodal power injection at a bus. A flow measurement device measures the real and reactive power flows of a branch.

Based on the measurement device deployment, a bus in a grid may be one of three possible types:

- A *PMU bus* is a bus where a PMU is placed at.
- An *injection bus* is a bus equipped with an injection measurement device. Since the injection measurements of a zero-injection bus (ZIB) are known (they are zero), we also refer to a ZIB as an injection bus.
- *Simple bus* is a bus that is neither a PMU bus nor an injection bus.

B. Observability Rules

System observability depends on the connectivity among the buses as well as on the location where measurement devices are placed. A power system is fully observable if all its buses are observable. A bus is said to be observable if the bus' state (voltage phasors) can be estimated from the set of available measurements. As stated above, the measurements can be phasor measurements from PMUs and conventional P-Q measurements (injection and flow measurements).

The set of observable buses is determined by applying the following rules [4], [6], [8], [20].

- 1) **Rule 1:** A bus is observable if it is a PMU bus.
- 2) **Rule 2:** A bus is observable if it is adjacent to at least one PMU bus.
- 3) **Rule 3:** For any set of buses made of an injection bus and all its adjacent buses, if we know that all but one bus in the set are observable, then all buses in the set are observable.
- 4) **Rule 4:** If a flow measurement (active and reactive) of a branch is known and if one of its terminal buses is observable, the other terminal bus is observable.

C. System Model and Problem Definition

We model a power system as an undirected graph on the set of vertices $B = \{1, 2, \dots, n\}$ that represent the buses. We define the set $P = \{1, 2, \dots, m\}$ as the set of PMUs deployed in the power system and $\beta : P \rightarrow B$ the mapping such that $\beta(j) = b$ when PMU $j \in P$ is placed at bus $b \in B$.

During the time a utility rolls out a software patch, a PMU in a grid is in one of the following three states:

- State (1): unpatched and streaming phasor measurement,
- State (2): being patched and offline,
- State (3): patched and streaming phasor measurement.

We assume that a state estimator receives and processes measurements from PMUs that are in state (1) as well as those in state (3) to compute the system state during the patching time window. Further, we assume that no PMU goes offline due to failure during the time a software patch is being rolled out.

A PMU patching problem is stated as finding a partitioning of deployed PMUs into as few disjoint groups as possible such that all the PMUs in one group can be transformed from State (1) through State (2) to State (3) in one round while the PMUs in all the other subsets along with any available conventional measurements provide full system observability during that round. Once such a partition is found, the patch is applied to all PMUs in as many rounds as there are subsets in the partition.

Since software patches for P-Q measurement devices and those for PMUs are likely to be different, a patching plan for the P-Q measurement devices is done separately (some P-Q devices may not even be patchable) and that is not a problem we address in this paper. Therefore, during the entire time PMUs are patched, we assume any available P-Q measurement devices remain operational.

III. THE SENSOR PATCHING PROBLEM (SPP)

In this section we give a set theoretic formulation of our problem, which we call the sensor patching problem (SPP). Further, we prove that it is NP-complete and give an ILP formulation.

A. Set Theoretic Formulation of SPP

Let $S = \{1, 2, 3, \dots, n\}$ is a finite set of sites to be observed and $P = \{1, 2, 3, \dots, m\}$ is a finite set of sensors that observe the sites. Further, let $\Gamma : S \rightarrow 2^P$ be a mapping such that $\Gamma(s)$ is the set of sensors in P that observe site $s \in S$, where 2^P is the set of all subsets (power set) of set P .

Definition 1. Given a non-empty finite set P , a k -tuple $\{c_1, c_2, c_3, \dots, c_k\}$ partitions set P if:

- $c_i \neq \emptyset, \forall i \in \{1, 2, \dots, k\}$.
- $\cup_{i=1}^k c_i = P$.
- $c_i \cap c_j = \emptyset$, for $1 \leq i < j \leq k$.

A feasible sensor patching plan is a partition $\{c_1, c_2, c_3, \dots, c_k\}$ of the set P such that the following observability condition is satisfied:

$$|\Gamma(s) \setminus c_i| \geq 1, \forall s \in S, \text{ and } i = 1, 2, \dots, k \quad (1)$$

Each subset c_i in the family of subsets that partition P defines the set of sensors that are patched at round i . A given sensor placement P has a feasible patching plan if and only if $|\Gamma(s)| \geq 2, \forall s \in S$, i.e., each site is observed by at least two sensors.

The sensor patching problem (SPP) is finding a sensor patching plan that minimizes k .

B. Mapping PMU Patching Problem to SPP

In this subsection, we describe how the PMU patching problem is mapped to the SPP problem. First we transform the topology and the constraints, in two steps, using the topology transformation and constraint modification methods described in [21].

1) Step 1: Merging an injection bus with a neighbor:

Observability condition (3) introduced in Section II-B enables us to transform the grid topology by merging an injection bus with any one of its neighboring buses and treat the transformed topology as an equivalent representation of the system when analyzing the system's observability. If there are multiple injection buses in the system, the merging processes is repeated until no such bus is left. Note that, in the modified topology, a bus can have more than one PMU if more than one of the merged buses were PMU buses. The bus merging operation does not affect the set of deployed PMUs. Hence, the set P is the same before and after the topology modification.

Once we modify the topology using the above merging technique, we obtain a new set of buses $B' = \{1, 2, 3, \dots, n'\}$, where $n' \leq n$. The set $\Gamma'(b')$ for $b' \in B'$ is defined as the set of PMUs placed at bus b' , if there are any, or in any of the buses that are adjacent to b' .

2) Step 2: Replacement of terminal buses of branches with flow measurements by another bus: Rule (4) in Section II-B states that if a branch is equipped with a flow measurement and one of its terminal buses is observable, the other terminal bus is also observable. Therefore, any two terminal buses $u, v \in B'$ whose connecting branch $u - v$ is equipped with a flow measurement are observable if and only if $\Gamma'(u) \cup \Gamma'(v) \neq \emptyset$. Likewise, if there are multiple branches equipped with flow measurements such that these branches and their terminal buses form a connected subgraph, all the terminal buses in the subgraph are observable if and only if $\forall u$ in the subgraph, $\cup \Gamma'(u) \neq \emptyset$. Using this observation, we further transform the topology of the grid by replacing all terminal buses in such a connected subgraph by a single bus α and by defining $\Gamma(\alpha) = \cup \Gamma'(u), \forall u$ in the connected subgraph to be the set of PMUs that make the bus α (i.e., all the buses in the original topology making up α) observable.

Figure 1 illustrates the two-step topology transformation. The original system in Figure 1(a) has $n = 9$ buses, with 1 injection bus (bus 5), two flow measurements at branches 2-3 and 3-4 and 4 PMUs at buses 1, 4, 6 and 7. In step 1, buses 5 and 8 are merged to form a new bus (bus 8 in Figure 1(b)). The resulting topology has $n' = 8$ buses. In step 2, the terminal buses of branches equipped with flow measurements (buses 2, 3 and 4 in Figure 1(b)) are replaced by a single bus (bus 2 in Figure 1(c)). The resulting topology has $n'' = 6$ buses.

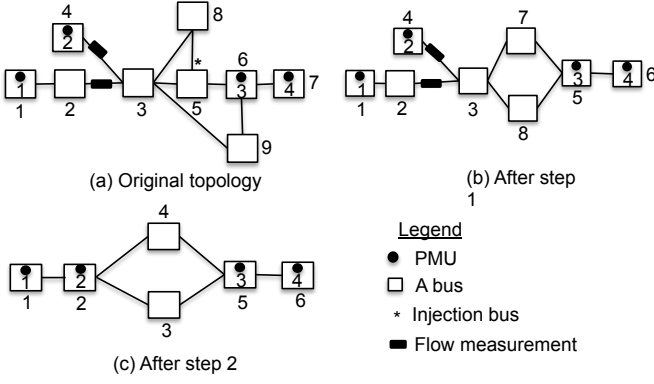


Fig. 1: Topology transformation for mapping to SPP.

After performing the topology transformation following the above techniques, we obtain a new set of buses $B'' = \{1, 2, 3, \dots, n''\}$ where $n'' \leq n'$, and new sets $\Gamma(b''), \forall b'' \in B''$ and the set of PMUs P . After this pre-processing, the PMU patching problem is directly mapped to the SPP by defining the set of buses $B'' = \{1, 2, 3, \dots, n''\}$ in the grid as the sites to be observed in the SPP and the set of deployed PMUs $P\{1, 2, 3, \dots, m\}$ as the set of sensors in SPP and the set $\Gamma(b'')$ of PMUs observing bus b'' as the set of sensors observing a site.

Note that a PMU patching plan has a feasible solution if and only if each bus in the transformed topology is observed by at least two PMUs, i.e., if $\forall b'' \in B'', |\Gamma(b'')| \geq 2$.

C. NP-completeness proof of SPP

Below, we show that the decision problem version of the SPP is NP-complete. The decision problem of SPP is defined as follows:

SPP Decision problem:

- **Instance:** Finite sets S and P , a mapping $\Gamma : S \rightarrow 2^P$ and an integer $k \geq 2$.
- **Question:** Is there a partitioning of the set P into at most k disjoint subsets $\{c_1, c_2, c_3, \dots, c_k\}$ such that the observability condition in Eq. (1) is satisfied?

Theorem 1. *The decision version of SPP is NP-complete.*

Proof. The first step of the proof is to show that SPP is in NP. Given a nondeterministically selected partition of P into k disjoint subsets, we can determine if the partition satisfies the observability condition in Eq. (1) in polynomial time. Hence SPP is in NP.

The second step of our proof is to select a known NP-complete problem and construct a polynomial time transformation that maps any instance of the NP-complete problem to an SPP problem. For our proof, we choose the hypergraph coloring problem (HCP), which is NP-complete.

A hypergraph is denoted by $H = (V, E)$, where V is a finite set of vertices and E is a set of hyperedges whose elements are subsets $e \subseteq V$ such that $\cup_{e \in E} e = V$. Given a hypergraph $H = (V, E)$ and an integer $k \geq 2$, a k -coloring of a hypergraph H is an allocation of colors to the vertices such that:

- A vertex has just one color.

- We use k colors to color all the vertices.
- No hyperedge with a cardinality more than one has all its vertices of the same color, i.e., no such hyperedge is monochromatic.

Any feasible coloring of a hypergraph using k colors induces a partition of the set of vertices V in k color classes: $\{c_1, c_2, c_3, \dots, c_k\}$ such that for $e \in E, |e| \geq 2$ then $e \not\subseteq c_i, \forall i \in \{1, 2, 3, \dots, k\}$ [22].

HCP Decision problem:

- **Instance:** Hypergraph $H = (V, E)$, an integer $k \geq 2$.
- **Question:** Is there a partitioning of the set of vertices V into at most k classes $\{c_1, c_2, c_3, \dots, c_k\}$ such that $\forall e \in E, |e| \geq 2, e \not\subseteq c_i, \forall i \in \{1, 2, 3, \dots, k\}$?

Having introduced HCP, let's now look at how to transform an instance of an HCP to an instance of SPP in polynomial time. Given an instance $HCP(V', E', k)$ where $|e'| \geq 2, \forall e' \in E'$, we construct an instance $SPP(P', S', \Gamma, k)$, where $P' \leftarrow V', S' \leftarrow E', k \leftarrow k, \Gamma \leftarrow Id_E$ such that $\Gamma : e' \rightarrow e', \forall e' \in E'$. This transformation from HCP to SPP is a polynomial time (trivial) transformation.

Assume we have an oracle that solves any given SPP decision problem. The oracle outputs “yes” to the instance $SPP(P', S', \Gamma, k)$ if and only if there exists a partition of P' to k subsets $\{c_1, c_2, c_3, \dots, c_k\}$ such that $(\Gamma(b') \setminus c_i) \neq \emptyset, \forall s' \in S', \forall i \in \{1, 2, 3, \dots, k\}$. Because of the mapping stated above, this is also the same as saying the oracle outputs “yes” if and only if $e' \not\subseteq c_i, \forall e' \in E', \forall i \in \{1, 2, 3, \dots, k\}$, which is the same as the “yes” output if there is a solution to the HCP decision problem. Therefore, if we can transform HCP to SPP and solve it, it means SPP is at least as hard as HCP. Hence, SPP is NP-complete.

By showing that the SPP is as hard as HCP, it also follows that even if we were told the set of sensors in a given instance of SPP could be patched in only two rounds, there is no efficient algorithm that can find any reasonable approximation for the number of rounds. \square

D. BILP Formulation of SPP

Now that we have shown SPP is NP-complete, we formulate it as a binary integer linear programming (BILP) minimization problem and use a BILP solver to find optimal solutions for small size networks and sub-optimal solutions for large network sizes.

To formulate SPP as a BILP problem, we use the *representatives* method introduced in [23]. As stated above, our goal is to find the minimum number of subsets $\{c_1, c_2, \dots, c_k\}$ that partition set P such that for any $s \in S$ the sensors in $\Gamma(s)$ cannot all be assigned to the same subset. The representatives formulation, as its name indicates, chooses one element from each of the partitioning subsets as a representative element to the subset (to all the elements in the subset). Therefore, each element in P can be in one of two states: either it represents the subset it is an element of or there exists another element that represents its subset. To describe this, we use an $m \times m$ matrix x of binary variables where $m = |P|$ is the number of sensors and the variables are defined by:

$$x_{i,j} = \begin{cases} 1 & \text{if element } i \text{ represents element } j, \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

Variable $x_{i,j}$ can be 1 only if elements i and j are in the same subset. By definition the representative elements are the elements i with $x_{i,i} = 1$. If $x_{i,i} = 1$, the row $x_{i,-}$ is an indicator vector of one of the subsets that partition the set P .

A BILP formulation of SPP is given as follows:

$$\min \quad \sum_{i=1}^m x_{i,i} \quad (3)$$

$$\text{s.t.} \quad \sum_{i=1}^m x_{i,j} = 1, \quad \forall j \in \{1, 2, \dots, m\} \quad (4)$$

$$\sum_{j \in \Gamma(s)} x_{i,j} \leq (|\Gamma(s)| - 1) \cdot x_{i,i}, \quad \forall s \in S, \quad \forall i \in \{1, 2, \dots, m\} \quad (5)$$

$$x_{i,j} \in \{0, 1\}, \quad \forall i, j \in \{1, 2, \dots, m\} \quad (6)$$

Claim 1. *A solution to the BILP problem 3 - 6 is an optimal solution to the SPP.*

Proof. Constraint (4) guarantees each sensor has only one representative. This is equivalent to saying each sensor is assigned to only one subset. This means two things: first, it means no two subsets can have a common element; second, the union of the subsets is P . Therefore, the subsets are feasible partitions of set P . Constraint (5) makes sure that the sensors in the set $\Gamma(s)$ cannot all choose the same representative sensor i and requires that $x_{i,i} = 1$ if sensor i is chosen as representative to one of the sensors in $\Gamma(s)$. This constraint guarantees that every bus has at least two of the sensors that observe it assigned to different subsets, i.e., the observability condition is satisfied. Constraint (6) states the variables are binary.

All the constraints represent the constraints for an SPP. Since the objective function (3) minimizes the number of representative sensors, which is the same as minimizing the number of subsets that partition P , the solution to the BILP problem is an optimal solution to the optimization version of SPP. \square

In Section VI, we solve the above BILP problem using the LP solver package *lp_solve* [24] for different bus systems.

IV. THE CASE OF RADIAL-STRUCTURED NETWORKS

In section III, we have seen that the general case (where the grid topology is a mesh network) is NP-complete. Therefore, the problem is in general solved using a heuristic approach. However, there is an important case when the topology of the grid has a radial structure (is a tree) where the problem can be optimally solved in polynomial time. The special case is of interest to us because the active configuration of many power distribution networks is a tree.

Theorem 2.

- 1) *Given a system model where the topology is a tree and the set of measurements from PMUs and P-Q measurement*

devices guarantees a feasible PMU patching plan, the minimum number of rounds required to patch all the PMUs is equal to 2.

- 2) *An optimal patching plan is given by Algorithm 1; its complexity is $O(|B''|^2)$, where B'' is the set of buses in the transformed topology obtained in Section III-B.*
- 3) *There exists a non-tree grid topology where the optimal PMU patching plan is more than 2.*

In the description of the algorithm, we phrase the problem as a two-coloring problem and say that two PMUs have the same color if they are allocated to the same round. We say c_0 [resp. c_1] is the set of PMUs that are assigned to the first [resp. second] round i.e., colored in, say, red [resp. blue].

The algorithm operates on the transformed topology obtained in Section III-B. We assume the initial topology is a tree, and it can easily be seen that the transformation preserves the tree property.

Therefore, the algorithm takes as input (i) the transformed tree of buses $B'' = \{1, 2, 3, \dots, n''\}$, (ii) for every bus $b'' \in B''$, the set $\Gamma(b'')$ of PMUs that make b'' observable and (iii) for every PMU $j \in P$ its location $\beta(j) \in B''$.

See Figures 2 and 3 for an illustration.

The proof of the theorem will make use of the following lemmas.

Lemma 1. *The algorithm's inputs have the following properties:*

- 1) $\forall b'' \forall j : \text{if } j \in \Gamma(b''), \text{ then } d(b'', \beta(j)) \leq 1, \text{ where } d() \text{ returns the shortest distance (number of edges) between two buses.}$
- 2) $\forall j \in P, j \in \Gamma(\beta(j)).$
- 3) $\forall j, j' \in \Gamma(b''), \beta(j) \neq b'', \beta(j') \neq b'', \text{ then } \beta(j) \neq \beta(j').$

Proof. Property (1) states that if a PMU j observes b'' , then either $\beta(j) = b''$ or $\beta(j)$ is adjacent to b'' . Let $\exists j' \in P$ such that $d(b'', \beta(j')) \geq 2$. This indicates that the shortest path between b'' and the bus where PMU j' was placed at in the original topology has at least two edges that were not equipped with a flow measurement and none of the terminals for these two edges are injection bus. By applying observability rules (1) and (2) in Section II-B, we can conclude that PMU j' cannot be in $\Gamma(b'')$. Therefore, if $j \in \Gamma(b'')$, then $d(b'', \beta(j)) \leq 1$. However, there may be some PMUs placed at a bus adjacent to b'' that are not in $\Gamma(b'')$.

Property (2) states if a PMU is placed at a bus, then the PMU observes the bus. For PMU $j \in P$, if $b'' = \beta(j)$, it means one of three possible cases: (1) b'' is the same bus where PMU j was placed at in the original topology; (2) b'' is a new bus formed by merging buses in step 1 of the topology transformation and one of the merged buses was where PMU j was placed at before the transformation; (3) b'' is a new bus formed in step 2 of the topology transformation and PMU j was in one of the buses forming this new bus. In all three cases, PMU j observes all the buses making up b'' by observability rule 1 (for case 1), observability rules 2 and 3 (for case 2) and observability rule 4 (for case 3).

Property (3) states a bus cannot have more than 2 PMUs that observe an adjacent bus. In the original topology, a maximum of one PMU is assumed to be placed at a bus. If there are two PMUs in a bus in the transformed topology observing another adjacent bus, then the two buses in the original topology where the PMUs were placed at and the adjacent bus being observed form a loop, which makes the original topology a non-tree. \square

Algorithm 1 Find a 2-round patching plan on a tree

Inputs: P, B'', Γ, β

Output: c_0, c_1

Steps

- 1) Select one bus $\rho \in B''$ and call it the root of the tree.
- 2) For each $j \in P$, color j according to its distance from the root $d(\beta(j), \rho)$ and build the color classes c_0 and c_1 as follows:

$$\forall i \in \{0, 1\}, c_i = \cup \{j : i = d(\beta(j), \rho) \bmod 2\} \quad (7)$$

- 3) **While** $\exists b'' \in B''$ that violates the condition:

$$\forall i \in \{0, 1\}, |\Gamma(b'') \setminus c_i| \geq 1 \quad (8)$$

- a) Select b'' with the maximum $d(b'', \rho)$ (breaking ties arbitrarily).
- b) **If** b'' is not a PMU bus:
 - i) Select a child bus u such that $u \in \cup \{\beta(i), \forall i \in \Gamma(b'')\}$.
 - ii) Flip the color of each PMU placed at the buses of the subtree rooted at u .
- c) **Else If** b'' has one or more PMUs placed at its location:
 - i) **If** $\exists j \in \Gamma(b'')$ such that $\beta(j) = u$ and u is a child of b'' :
 - A) Flip the color of all PMUs in the subtree rooted at u .
 - ii) **Else**
 - A) Select PMU $j \in \Gamma(b'') : \beta(j) = b''$.
 - B) Flip the color of the selected PMU j .
 - C) For each $u \in B'' : j \in \Gamma(u)$ and u is a child bus of b'' , flip the color of all PMUs in the subtree rooted at u .

- 4) **End while**

Lemma 2. If bus b'' that violates the condition in Eq. (8) is not a PMU bus, then the bus always has, at least, one child bus u with a PMU that is an element of $\Gamma(b'')$, i.e. Step 3(b)i in the algorithm is well-defined.

Proof. By necessity of the PMU patching problem, $|\Gamma(b'')| \geq 2$. Property (3) in Lemma 1 states that a bus cannot have two PMUs that observe an adjacent bus. Therefore, if a violating bus b'' is not a PMU bus, at most one of the PMUs in $\Gamma(b'')$ can be placed at b'' parent bus. The remaining PMUs in $\Gamma(b'')$ must, therefore, be placed at one or more of its children buses. \square

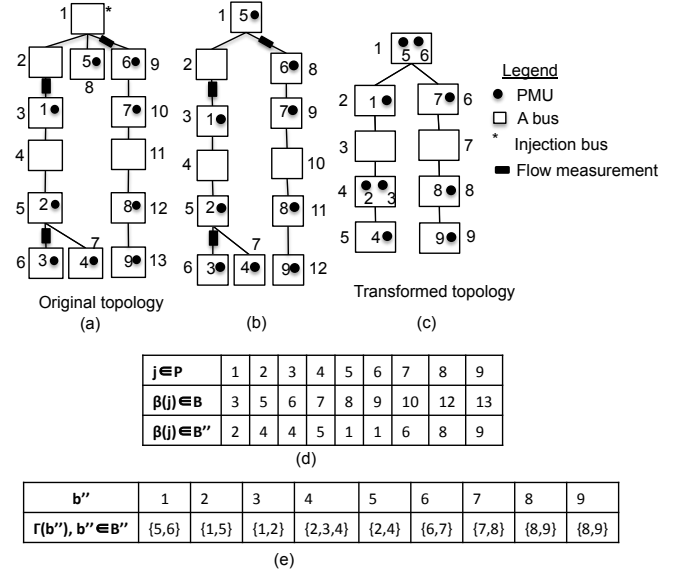


Fig. 2: A topology transformation of a 13-bus power system that deploys 9 PMUs, 3 flow measurements and one injection measurement. (a) Original topology. (b) topology transformation after merging injection bus with a neighbor. (c) topology transformation after replacing a group of terminal buses of flow measurements by a single bus. (d) PMU IDs and their corresponding locations in the original topology and in the transformed topology. (e) set $\Gamma(b'')$ for every b'' in the transformed topology.

Lemma 3. If a child bus u of a violating bus b'' has a PMU $j \in \Gamma(b'')$ and no bus in the subtree rooted at u contains a violating PMU, then flipping the color of PMUs in the subtree does not cause any PMU in the subtree to violate, i.e., Steps 3(b)ii and 3(c)iA in the algorithm don't cause a PMU in the subtree to violate.

Proof. Let T_u denote the subtree rooted at the child bus u . Any bus in T_u except, possibly, the root u is observed only by PMUs placed at buses in T_u (by Property (1) of Lemma 1). Similarly, if the violating bus b'' is not a PMU bus, all PMUs in $\Gamma(u)$ are also placed at buses in T_u .

If the violating bus b'' has one or more PMUs placed at it, there may be a PMU $j' \in \Gamma(u)$ such that $\beta(j') = b''$. Since b'' is a violating bus, PMU j' must have the same color as the PMU $j \in \Gamma(b'')$, where $\beta(j) = u$. Since u is not a violating bus, there must be another PMU j''' in $\Gamma(u)$ whose color is different from that of j . Because of Property (3) of Lemma 1, $\beta(j''') \neq b''$; thus $\beta(j''')$ must be a bus in T_u , i.e., $|\Gamma(u)| \geq 3$. Thus, u would not violate the condition irrespective of the existence of PMU j' at b'' .

Since we have shown, in all cases, that the coloring of PMUs in T_u guarantees all buses in the subtree (including u) don't violate the condition, flipping the colors of the PMUs placed at any bus in T_u does not cause any bus in T_u to violate the condition. \square

We now proceed with the proof of Theorem 2.

Proof. A. The first iteration of the algorithm has three possible cases: (1) There is no violating bus in the tree. In this case, it means that the initial coloring based on the distance from the root guarantees that the set $\Gamma(b''), \forall b'' \in B''$ is not monochromatic. These the algorithm terminates with a valid coloring of each PMU using only two colors. (2) There is a

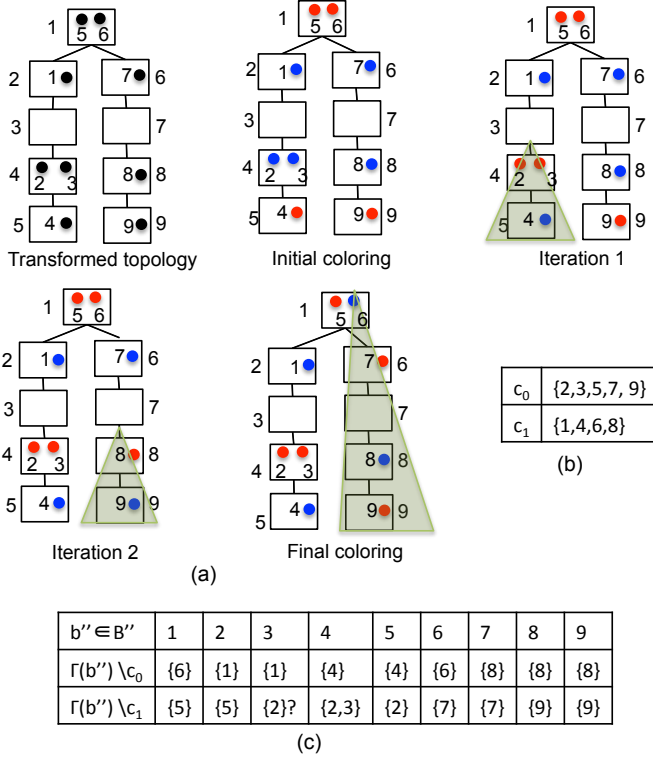


Fig. 3: Application of Algorithm 1 on a 13-bus system that deploys 9 PMUs, 3 flow measurements and one injection measurement such that one subset of PMUs can provide full observability while the other subset of PMUs is being patched. (a) coloring of PMUs at different iterations. (b) The two color sets c_0 (red) and c_1 (blue) when the algorithm stops. (c) Each bus is observable by at least 1 PMU in one subset when the other subset of PMUs is being patched.

violating leaf bus b'' at the lowest possible level. By design of the initial coloring, a leaf node b'' violates the condition at the first iteration if and only if $\forall j \in \Gamma(b''), \beta(j) = b''$. The algorithm will, therefore, execute Step 3(c)iiB. By flipping the color of one bus in $\Gamma(b'')$, we ensure bus b'' no more violates the condition and end up with a one-bus subtree that is not violating. (3) There is a non-leaf violating bus b'' at the lowest possible level. If b'' has a PMU $j \in \Gamma(b'')$ such that $\beta(j)$ is a child bus of b'' , then b'' is made non-violating by flipping the color of PMU j (using Step 3(b)ii or Step 3(c)iA in the algorithm). If there is no such a PMU, then b'' is made observable by flipping the color of one of the PMU's placed at its location (using Step 3(c)iiB in the algorithm). This guarantees b'' has one PMU which has a different color from the rest of PMUs in $\Gamma(b'')$, thus making it non-violating.

By Lemma 3, Steps 3(b)ii, 3(c)iA and 3(c)iiC don't cause any bus in the subtrees to be violating at the end of the iteration. Hence, the first iteration ends by fixing one violating bus b'' in the tree and resulting is one subtree rooted at b'' that is guaranteed to have no violating bus during the subsequent iterations.

B. Like the first iteration, every subsequent iteration of the algorithm transforms one violating bus to a non-violating one. Since there can be only a finite number of violating buses after the initial coloring and since we have shown that every iteration in the algorithm removes one violating bus without introducing any new violating bus in the so-far-explored parts

of the tree, the algorithm eventually terminates after fixing all the violating buses. The final coloring partitions the set of PMUs into two disjoint color classes. Consequently, all the PMUs can be patched in only two rounds by patching PMUs in one color class in the first round and those in the other color class in the second round. This proves the first point of Theorem 2.

In the initial coloring, the maximum possible number of violating vertices is $|B''|$. Since each iteration in our algorithm fixes only one violating bus, all violating vertices are fixed in a maximum of $|B''|$ iterations. Each iteration runs in linear time because the maximum possible number of vertices in any subtree T_u is $|B''|$. Hence the complexity of the algorithm to obtain an optimal coloring is $O(|B''|^2)$, which proves the second point of Theorem 2.

The third point of Theorem 2 is proved by the 3-round optimal patching plan for the IEEE 57-bus system obtained by using the ILP solver as shown in Table I. \square

V. APPROXIMATION ALGORITHM FOR NON-TREE CASES

It is common to model NP-complete problems as ILP problems and use ILP solvers to find optimal or suboptimal solutions for a relatively small size of input. However, ILP solvers tend to be too slow to find even a suboptimal solution as the input size grows. The alternative is to use heuristic algorithms that find approximate solutions much faster than ILP solvers. For this reason, we propose a heuristic algorithm that finds an approximate solution to the SPP, which we have shown to be NP-complete.

A. A Greedy Heuristic Algorithm

Before going to the details of the heuristic algorithm, let's first define the collection $\mathcal{O} = \{o_j : j \in P\}$, where o_j is the observability set of PMU j defined as:

$$o_j = \{b'' : b'' \in B'', j \in \Gamma(b'')\} \quad (9)$$

In other words, o_j is the set of buses that are made observable by PMU j .

Given the set of all buses in the transformed topology $B'' = \{1, 2, 3, \dots, n''\}$, the set of deployed PMUs P and the collection \mathcal{O} , the heuristic algorithm we propose follows a greedy approach that maximizes the set of PMUs that are patched at each round while still maintaining full system observability. Given a set of unpatched PMUs P , finding the maximum number of PMUs to patch is equivalent to finding the minimum number of PMUs that provide full system observability, which is exactly the same as solving the minimum set cover (MSC) problem over a universe B'' and a collection of subsets \mathcal{O} . The set of PMUs to patch is, therefore, the set that contains the PMUs that are not in the MSC solution. Once these PMUs are patched, they will resume streaming for the rest of the time. Therefore, the observability condition for the set of buses that are in the observability sets of these PMUs will always be satisfied for the remaining patch rounds. Hence, before we select the next set of PMUs to patch, we perform the following preprocessing:

- Remove all the observability sets of all the already patched PMUs from \mathcal{O} .
- Remove all the buses in the observability sets of the already patched PMUs from the universe B'' .
- Remove all the buses in the observability sets of the already patched PMUs from the observability set of the yet unpatched PMUs.

After the pre-processing, we proceed with the same greedy approach (solving the MSC problem) for the updated universe B'' and the updated collection \mathcal{O} . We repeat this process until $B'' = \emptyset$ (until all buses are observed by the already patched PMUs). At this stage, if there are still any PMUs that are not yet patched ($\mathcal{O} \neq \emptyset$), we patch all such PMUs at once in the final round. Algorithm 2 shows the pseudocode for our heuristic algorithm. The algorithm outputs a collection $\mathcal{C} = \{c_1, c_2, \dots, c_k\}$, where $c_i \subset P$ is the set of PMUs patched in round i and k is the total number of rounds required to patch all the PMUs.

Since the MSC problem is itself NP-complete, we use the most commonly used greedy heuristic to solve it. The greedy heuristic for MSC chooses the subset that maximizes the number of new elements in the universe B'' that are not yet covered by the already selected subsets.

B. Formulation as Submodular Maximization

Here we show that the SPP can be formulated as a maximization of a submodular set function. It is known that a greedy heuristic guarantees a reasonably good approximation to the optimal solution for problems that are submodular and the SPP is in that category. This may be used as a justification to why the greedy algorithm proposed above can be expected to perform well.

Given the set of PMU's P and $m = |P|$, let's define the collection Ψ as:

$$\Psi = \{\psi : \psi \subseteq P, \cup_{j \in (P \setminus \psi)} o_j = B''\} \quad (10)$$

In other words, an element in Ψ is a set of PMUs that can be taken offline and full system observability can still be maintained. From this, it follows that if $\mu \in \Psi$ and $\mu' \subset \mu$, then $\mu' \in \Psi$.

Consider a non-negative submodular set function $f : 2^\Psi \rightarrow \mathbb{R}_+$ on Ψ that assigns a non-negative number to every subset of the set Ψ .

Claim 2. A collection $\mathcal{C} \subset \Psi$ that maximizes the following set function,

$$f(\mathcal{C}) = Q \cdot |\cup_{c \in \mathcal{C}} c| - |\mathcal{C}|, \text{ where } Q > m \text{ is a constant.} \quad (11)$$

is an optimal solution to the SPP.

Proof. Let $\mathcal{C}^* = \{c_1, c_2, \dots, c_{k^*}\}$ be an optimal solution to the SPP. Passing \mathcal{C}^* as an input to f , we get.

$$f(\mathcal{C}^*) = Q \cdot |\cup_{c \in \mathcal{C}^*} c| - |\mathcal{C}^*| = -k^* + Q \cdot m \quad (12)$$

We want to show that $f(\mathcal{C}) < -k^* + Q \cdot m$ for any input $\mathcal{C} = \{c_1, c_2, \dots, c_k\}$, where $k > k^*$.

Algorithm 2 Partition P into minimum patchable subsets using a greedy heuristic

```

1: Input:  $\mathcal{O}, B''$ 
2: Output:  $\mathcal{C} := \{c_1, c_2, \dots, c_k\}$ 
3:  $round = 1$ 
4:  $\mathcal{C} := \{\}$ 
5: while  $B'' \neq \emptyset$  do
6:    $\sigma := \text{FindMSC}(\mathcal{O}, B'')$ 
7:    $c_{round} := \{j : o_j \notin \sigma\}$ 
8:    $\mathcal{C} := \mathcal{C} \cup \{c_{round}\}$ 
9:    $\mathcal{O} := \mathcal{O} \setminus \{o_j : j \in c_{round}\}$ 
10:   $B'' := B'' \setminus \{\cup o_j : j \in c_{round}\}$ 
11:  for  $o_u \in \mathcal{O}$  do
12:     $o_u := o_u \setminus \{\cup o_j : j \in c_{round}\}$ 
13:  end for
14:   $round++$ 
15: end while
16: if  $\mathcal{O} \neq \emptyset$  then
17:    $c_{round} := \{j : o_j \in \mathcal{O}\}$ 
18:    $\mathcal{C} := \mathcal{C} \cup \{c_{round}\}$ 
19: end if

20: procedure  $\text{FindMSC}(\mathcal{O}, B'')$ 
21:    $\Delta := \emptyset$ 
22:    $\sigma := \emptyset$ 
23:   while  $\Delta \neq B''$  do
24:      $MaxCount := 0$ 
25:      $idx = 0$ 
26:     for all  $o_j \in \mathcal{O}$  do
27:       if  $|o_j \setminus \Delta| > MaxCount$  then
28:          $MaxCount := |o_j \setminus \Delta|$ 
29:          $idx := j$ 
30:       end if
31:     end for
32:      $\Delta := \Delta \cup o_{idx}$ 
33:      $\sigma := \sigma \cup \{o_{idx}\}$ 
34:   end while
35:   Return  $\sigma$ 
36: end procedure

```

Given a collection $\mathcal{C} = \{c_1, c_2, \dots, c_k\}$ for some $k \geq 1$,

$$f(\mathcal{C}) = -k + Q \cdot |\cup_{c \in \mathcal{C}} c| \quad (13)$$

Let $m' = |\cup_{c \in \mathcal{C}} c|$

$$\begin{aligned} f(\mathcal{C}) &= -k + Q \cdot m' \leq -k + Q \cdot (m - 1) \\ f(\mathcal{C}) &\leq -k + Q \cdot (m - 1) \end{aligned} \quad (14)$$

Since $Q > m$ and $k^* < k \leq m$, it is easy to show that

$$-k + Q \cdot (m - 1) < -k^* + Q \cdot m \quad (15)$$

Therefore,

$$\begin{aligned} f(\mathcal{C}) &\leq -k + Q \cdot (m - 1) < -k^* + Q \cdot m \\ f(\mathcal{C}) &< -k^* + Q \cdot m, \quad \forall \mathcal{C} \text{ where } |\mathcal{C}| > k^* \end{aligned} \quad (16)$$

This means,

$$\max_{\mathcal{C} \in \Psi} f(\mathcal{C}) = -k^* + Q \cdot m \quad (17)$$

which is the same as the optimal solution for SPP. \square

Claim 3. The set function f in Eq. (11) is submodular.

Proof. Function f is submodular if for all subsets $Y \subset X \subset \Psi$ and all $\mu \in \Psi \setminus X$,

$$f(Y \cup \{\mu\}) - f(Y) \geq f(X \cup \{\mu\}) - f(X) \quad (18)$$

We want to see if this holds true for f given in Eq. (11),

$$\begin{aligned} Q \cdot (|\cup_{y \in Y} y \cup \mu| - |Y \cup \{\mu\}| - Q \cdot (|\cup_{y \in Y} y|) + |Y|) &\stackrel{?}{\geq} \\ Q \cdot (|\cup_{x \in X} x \cup \mu| - |X \cup \{\mu\}| - Q \cdot (|\cup_{x \in X} x|) - |X|) &\quad (19) \end{aligned}$$

Using the substitutes $\mathbb{Y} = \cup_{y \in Y} y$ and $\mathbb{X} = \cup_{x \in X} x$, we get

$$\begin{aligned} Q \cdot |\mathbb{Y} \cup \mu| - |\mathbb{Y}| - 1 - Q \cdot |\mathbb{Y}| + |\mathbb{Y}| &\stackrel{?}{\geq} \\ Q \cdot |\mathbb{X} \cup \mu| - |\mathbb{X}| - 1 - Q \cdot |\mathbb{X}| + |\mathbb{X}| &\quad (20) \end{aligned}$$

$$|\mathbb{Y} \cup \mu| - |\mathbb{Y}| \stackrel{?}{\geq} |\mathbb{X} \cup \mu| - |\mathbb{X}| \quad (21)$$

$$|\mathbb{Y} \cup \mu| + |\mathbb{X}| \stackrel{?}{\geq} |\mathbb{X} \cup \mu| + |\mathbb{Y}| \quad (22)$$

$$|(\mathbb{Y} \cup \mu) \cup \mathbb{X}| + |(\mathbb{Y} \cup \mu) \cap \mathbb{X}| \stackrel{?}{\geq} |\mathbb{X} \cup \mu| + |\mathbb{Y}| \quad (23)$$

$$|\mu \cup \mathbb{X}| + |\mathbb{Y} \cup (\mu \cap \mathbb{X})| \stackrel{?}{\geq} |\mathbb{X} \cup \mu| + |\mathbb{Y}| \quad (24)$$

$$|\mathbb{Y} \cup (\mu \cap \mathbb{X})| \stackrel{?}{\geq} |\mathbb{Y}| \quad (25)$$

If $\mu \cap \mathbb{X} = \emptyset$, the right hand side of Eq. (25) is the same as the left hand side. Otherwise, the right hand side is strictly greater than the left hand side. Hence, f is submodular. \square

VI. SIMULATION RESULTS AND COMPARISONS

We compare the performance of our heuristic algorithm to results obtained from an ILP solver for different feeder bus systems. In order to make the comparison, we first find an optimal PMU placement that has a feasible patching plan. While determining an optimal PMU placement, we consider only measurements from PMUs while determining the system's observability. Besides, we do not use the observability rule that exploits the presence of zero-injection buses. Therefore, we solve the following simplified BILP minimization problem to determine the optimal PMU placement:

$$\min \sum_{j=1}^n p_j \quad (26)$$

$$\sum_{j=1}^n a_{i,j} \cdot p_j \geq 2 \quad (27)$$

$$p_i \in \{0, 1\} \quad (28)$$

where

$$a_{i,j} = \begin{cases} 1 & \text{if bus } i = j \text{ or } i \text{ is incident to } j \\ 0, & \text{otherwise} \end{cases} \quad (29)$$

The set $\{p_i : i \in B\}$ is a set of binary variables such that $p_i = 1$ if a PMU is placed in bus i and $p_i = 0$ otherwise. Solving the above BILP problem returns a placement with the fewest number of PMUs such that each bus is observed by at least two PMUs. That means it is guaranteed that such a placement has a feasible patching plan. Figure 4 shows an optimal PMU placement obtained by solving the above BILP for the IEEE 14-bus system.

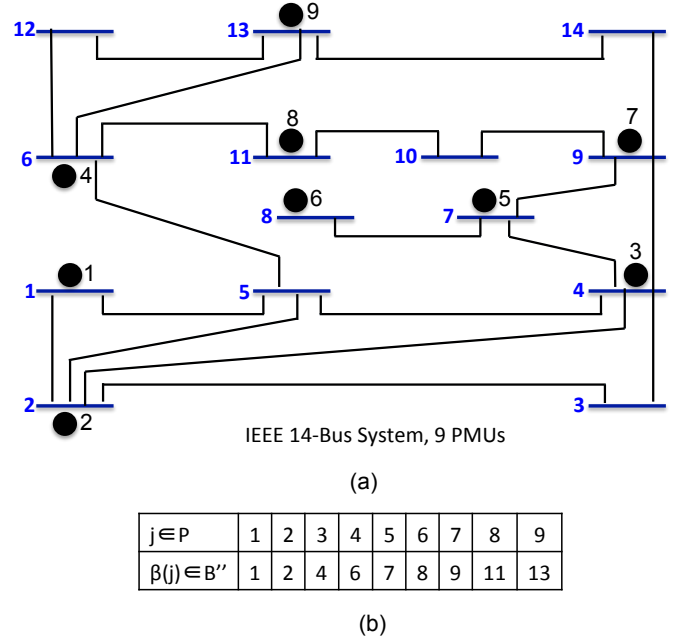


Fig. 4: Optimal PMU placement on the IEEE 14-bus system.

We use the open source ILP solver *lp_solve* [24] to solve the above BILP as well as the BILP formulation for the SPP introduced in Section III-D. We use a laptop with an Intel 2.8 GHz Core *i7* processor and 8GB RAM running Ubuntu 12.04 with Linux 3.2 for the simulations.

Our results in Table I show the optimal PMU placement and the number of rounds obtained both from the ILP solver and from the heuristic algorithm for different bus systems. The 4941-bus system represents the power transmission grid covering much of the western states in the United States as presented in [25]. The system has 4941 buses and 6594 branches. The data set for the bus system was obtained from [26]. The 189-bus system¹ represents Iceland's transmission network. It has 189 buses and 206 branches. All the other bus systems used in the simulation are the standard IEEE bus systems².

The simulation results show that the ILP performs better than the greedy algorithm in terms of finding fewer (optimal) number of rounds for small size networks. The ILP results for the 57-Bus system shows that the optimal number of rounds required is 3. This is one example that proves, unlike tree-structured grids, there are non-tree grids where the optimal patching plan is more than 2.

¹<http://www.maths.ed.ac.uk/optenergy/NetworkData/>

²<http://www2.ee.washington.edu/research/pstca/>

TABLE I: Performance comparison of ILP solver and a greedy algorithm for PMUs' patching plan.

Bus system	PMU Placement		Patching plan from ILP Solver			Greedy Patching Plan		
	#PMUs	PMU buses	Patchable groups of PMU buses	#R	Exec. time (sec.)	Patchable groups of PMU buses	#R	Exec. time (sec.)
14-Bus	9	[1, 2, 4, 6, 7, 8, 9, 11, 13]	[1, 4, 7, 11, 13] [2, 6, 8, 9]	2	0.014	[2, 8, 11, 13] [1, 6, 7, 9] [4]	3	0.004
30-Bus	21	[1, 2, 3, 5, 6, 8, 9, 10, 11, 12, 13, 15, 16, 18, 19, 21, 24, 25, 26, 27, 29]	[1, 2, 6, 9, 13, 15, 16, 19, 21, 25, 29] [3, 5, 8, 10, 11, 12, 18, 24, 26, 27]	2	0.127	[3, 5, 8, 11, 13, 16, 19, 21, 24, 26, 29] [1, 2, 6, 9, 10, 15, 18, 25, 27] [12]	3	0.005
39-Bus	28	[2, 3, 6, 8, 9, 10, 11, 13, 14, 16, 17, 19, 20, 22, 23, 25, 26, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39]	[2, 6, 9, 11, 14, 17, 19, 20, 22, 23, 25, 29, 32] [3, 8, 10, 13, 16, 26, 30, 31, 33, 34, 35, 36, 37, 38, 39]	2	0.399	[8, 11, 14, 17, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39] [3, 6, 9, 10, 13, 19, 20, 22, 23, 25, 29] [2, 16, 26]	3	0.012
57-Bus	33	[1, 2, 4, 6, 9, 11, 12, 15, 19, 20, 22, 24, 25, 26, 28, 29, 30, 32, 33, 34, 36, 37, 38, 39, 41, 44, 46, 47, 50, 51, 53, 54, 56]	[1, 6, 19, 22, 32, 36, 39, 41, 44, 51] [4, 9, 11, 15, 20, 24, 25, 28, 34, 37, 47, 50, 53, 56] [2, 12, 26, 29, 30, 33, 38, 46, 54]	3	340	[2, 6, 12, 19, 22, 28, 30, 33, 34, 39, 41, 44, 47, 51, 54] [1, 4, 9, 11, 20, 25, 26, 32, 37, 46, 50, 53, 56] [15, 24, 29, 36, 38]	3	0.016
118-Bus†	68	[1, 2, 5, 6, 9, 10, 11, 12, 15, 17, 19, 21, 22, 24, 25, 26, 27, 28, 29, 32, 34, 35, 37, 40, 41, 44, 45, 46, 49, 50, 51, 52, 54, 56, 59, 62, 64, 65, 66, 68, 70, 71, 73, 75, 76, 77, 78, 80, 83, 85, 86, 87, 89, 90, 92, 94, 96, 100, 101, 105, 106, 108, 110, 111, 112, 114, 116, 117]	[2, 5, 10, 12, 22, 24, 27, 28, 32, 34, 37, 41, 45, 49, 52, 56, 62, 64, 73, 75, 77, 80, 85, 87, 90, 94, 101, 105, 110, 116] [1, 6, 9, 11, 17, 21, 25, 29, 35, 40, 44, 46, 50, 51, 54, 59, 65, 66, 68, 70, 71, 76, 78, 83, 86, 89, 92, 96, 100, 106, 108, 111, 112, 114, 117] [15, 19, 26]	3	500	[2, 6, 10, 15, 19, 22, 26, 29, 35, 41, 44, 46, 54, 56, 65, 66, 73, 76, 78, 83, 87, 90, 96, 101, 106, 108, 111, 112, 114, 116, 117] [1, 9, 11, 12, 21, 27, 28, 32, 34, 40, 45, 50, 52, 62, 64, 71, 75, 77, 80, 86, 89, 94, 105, 110] [5, 17, 25, 37, 49, 51, 59, 68, 70, 85, 92, 100] [24]	4	0.125
189-Bus†	160	[1, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 24, 26, 27, 28, 29, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 53, 54, 55, 58, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 94, 95, 96, 97, 98, 99, 100, 101, 102, 104, 105, 106, 107, 108, 110, 111, 112, 113, 114, 115, 117, 118, 120, 121, 122, 123, 124, 126, 127, 128, 129, 130, 131, 134, 135, 136, 137, 138, 139, 141, 142, 143, 144, 145, 147, 150, 151, 152, 153, 154, 155, 156, 157, 159, 160, 162, 163, 164, 165, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189]	[1, 9, 13, 14, 15, 17, 24, 26, 31, 39, 40, 42, 45, 49, 60, 62, 66, 69, 71, 72, 78, 81, 82, 84, 88, 91, 94, 98, 100, 101, 104, 106, 108, 110, 114, 115, 117, 120, 126, 127, 130, 134, 138, 141, 144, 145, 147, 156, 159, 162, 168, 169, 174, 175, 176, 178, 179, 180, 181, 184, 186, 187] [5, 8, 10, 12, 16, 18, 19, 20, 21, 27, 28, 29, 32, 33, 34, 36, 37, 38, 41, 44, 47, 48, 50, 51, 52, 53, 54, 58, 61, 63, 64, 67, 68, 70, 73, 76, 77, 79, 83, 85, 86, 89, 96, 97, 102, 105, 111, 112, 113, 122, 123, 128, 129, 131, 136, 137, 139, 142, 143, 150, 151, 154, 155, 157, 160, 164, 165, 167, 172, 173, 182, 183, 185, 189] [4, 6, 7, 35, 43, 55, 65, 74, 80, 87, 90, 95, 99, 107, 118, 121, 124, 135, 152, 153, 163, 170, 171, 188]	3	500	[4, 6, 8, 9, 10, 13, 14, 16, 18, 19, 20, 21, 27, 28, 29, 32, 33, 34, 36, 37, 38, 42, 43, 45, 47, 48, 50, 51, 52, 53, 54, 55, 58, 61, 63, 65, 67, 68, 70, 74, 77, 79, 80, 81, 83, 85, 87, 89, 91, 95, 96, 98, 99, 100, 101, 102, 106, 107, 108, 111, 112, 114, 115, 117, 121, 122, 124, 128, 129, 131, 135, 136, 138, 142, 144, 145, 147, 150, 151, 154, 157, 160, 164, 165, 168, 169, 171, 172, 174, 175, 176, 178, 179, 180, 181, 182, 183, 186, 187, 189] [5, 7, 12, 15, 17, 31, 35, 39, 40, 41, 44, 49, 60, 62, 64, 66, 69, 73, 76, 78, 82, 84, 86, 88, 90, 94, 97, 104, 105, 113, 118, 120, 126, 127, 130, 134, 139, 141, 152, 153, 156, 159, 162, 163, 170, 173, 184, 185, 188] [1, 24, 26, 71, 72, 110, 123, 137, 143, 155, 167]	3	0.438
4941-Bus‡	3468	Not enough space to display	ILP solver did not converge	NA	NA	Not enough space to display	4	7716

#R:= number of rounds

†The ILP results for 118-bus and for 189-bus systems are sub-optimal. The simulation was stopped after 500 seconds.

‡The ILP solver does not have enough memory space to solve the 4941-bus system.

As the network size increases, the execution time for the ILP solver quickly increases and the resulting solution is sub-optimal. The execution times for the 118-bus and the 189-bus systems are too large that we had to stop the executions after 500 seconds forcing the solver to return sub-optimal solutions of 3 rounds each. Similarly, the memory requirement for the 4941-bus system is too large that the ILP solver could not solve it even sub-optimally using the machine we used for the simulation. Although the greedy approach does not find the optimal patching plan even for the small size networks, it finds a sub-optimal solution much faster. It also solves the 4941-bus system and finds a total number of rounds equal to only 4 within 7716 seconds. One can only imagine how slow an ILP solver can be to solve this problem even if the machine had enough memory size.

Table I shows the ILP solver finds a 2-round patching plan for a PMU placement on a 14-bus system (shown in Figure 4) where the set of buses whose PMUs will be patched in the first round is $r_0 = \{1, 4, 7, 11, 13\}$ and in the second round $r_1 = \{2, 6, 8, 9\}$, which can be easily observed from Table I. Likewise, the greedy algorithm finds a 3-round patching plan

TABLE II: Patching plan for the IEEE 14-bus system. The table shows the sets $\Gamma(b'')$, $\forall b'' \in B''$, the set of PMUs in $\Gamma(b'')$ that are operational at a given round of the ILP solver and the set of PMUs in $\Gamma(b'')$ that are operational at a given round of the greedy algorithm.

$b'' \in B''$	$\Gamma(b'')$	ILP solver rounds		Greedy algorithm rounds		
		$\Gamma(b'') \setminus r_0$	$\Gamma(b'') \setminus r_1$	$\Gamma(b'') \setminus r_0$	$\Gamma(b'') \setminus r_1$	$\Gamma(b'') \setminus r_2$
1	{1, 2}	{2}	{1}	{1}	{2}	{1, 2}
2	{1, 2, 3}	{2}	{1, 3}	{1, 3}	{2, 3}	{1, 2}
3	{2, 3}	{2}	{3}	{3}	{2, 3}	{2}
4	{2, 3, 5, 7}	{2, 7}	{3, 5}	{3, 5, 7}	{2, 3}	{2, 5, 7}
5	{1, 2, 3, 4}	{2, 4}	{1, 3}	{1, 3, 4}	{2, 3}	{1, 2, 4}
6	{4, 9}	{4}	{9}	{4}	{9}	{4, 9}
7	{3, 5, 6, 7}	{6, 7}	{3, 5}	{3, 5, 7}	{3, 6}	{5, 6, 7}
8	{5, 6}	{6}	{5}	{5}	{6}	{5, 6}
9	{3, 5, 7}	{7}	{3, 5}	{3, 5, 7}	{3}	{5, 7}
10	{7, 8}	{7}	{8}	{7}	{8}	{7, 8}
11	{4, 8}	{4}	{8}	{4}	{8}	{4, 8}
12	{4, 9}	{4}	{9}	{4}	{9}	{4, 9}
13	{4, 9}	{4}	{9}	{4}	{9}	{4, 9}
14	{7, 9}	{7}	{9}	{7}	{9}	{7, 9}

where the set of buses whose PMUs will be patched in the first round is $r_0 = \{2, 8, 11, 13\}$, in the second round $r_1 = \{1, 6, 7, 9\}$ and in the third round $r_2 = \{4\}$. In Table II, we

demonstrate that the system remains fully observable during all patching rounds - both for the ILP solver and for the greedy algorithm.

It is important to remember that a patching plan obtained using either of the methods can be re-used only if the network setting remains static. If there is change either in the PMU placement or in the connectivity among the buses, a utility needs to re-compute the patching plan for the new setting.

VII. CONCLUSION

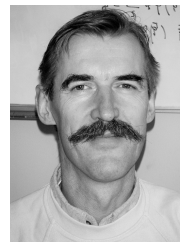
We have studied the PMU patching problem that arises when a utility wants to maintain system observability while applying software patches to PMUs. We consider a system that deploys both PMUs and conventional measurement devices. We have formulated the problem as a set theoretic problem which we proved to be NP-complete. We have also shown an interesting case, when the power grid is a tree, for which a polynomial time algorithm that computes an optimal patching plan that patches all the PMUs in only two rounds.

REFERENCES

- [1] S. Tom, D. Christiansen, and D. Berrett, "Recommended practice for patch management of control systems," *DHS control system security program (CSSP) Recommended Practice*, 2008.
- [2] K. Knorr, *Securing Critical Infrastructures and Critical Control Systems: Approaches for Threat Protection*. IGI Global, 2013, ch. Patching our Critical Infrastructure - Towards an Efficient Patch and Update Management for Industrial Control Systems.
- [3] M. Souppaya and K. Scarfone, "Guide to enterprise patch management technologies," *NIST Special Publication*, vol. 800, p. 40, 2013.
- [4] E. Abiri, F. Rashidi, and T. Niknam, "An optimal pmu placement method for power system observability under various contingencies," *International Transactions on Electrical Energy Systems*, vol. 25, no. 4, pp. 589–606, 2015.
- [5] T. L. Baldwin, L. Mili, M. B. Boisen, and R. Adapa, "Power system observability with minimal phasor measurement placement," *IEEE Transactions on Power Systems*, vol. 8, May 1993.
- [6] F. Aminifar, A. Khodaei, M. Fotuhi-Firuzabad, and M. Shahidepour, "Contingency-constrained pmu placement in power networks," *IEEE Transactions on Power Systems*, vol. 25, Feb 2010.
- [7] J. G. Philip and T. Jain, "Optimal placement of pmus for power system observability with increased redundancy," in *2015 Conference on Power, Control, Communication and Computational Technologies for Sustainable Growth (PCCCTSG)*, Dec 2015.
- [8] M. Esmaili, K. Gharani, and H. A. Shayanfar, "Redundant Observability PMU Placement in the Presence of Flow Measurements Considering Contingencies," *IEEE Transactions on Power Systems*, Nov 2013.
- [9] A. Enshaee, R. A. Hooshmand, and F. H. Fesharaki, "A new method for optimal placement of phasor measurement units to maintain full network observability under various contingencies," *Electric Power Systems Research*, vol. 89, 2012.
- [10] G. N. Korres, N. M. Manousakis, T. C. Xygkis, and J. Lfberg, "Optimal phasor measurement unit placement for numerical observability in the presence of conventional measurements using semi-definite programming," *IET Generation, Transmission Distribution*, vol. 9, no. 15, pp. 2427–2436, 2015.
- [11] W. Bao, R.-P. Guo, Z.-X. Han, L.-Y. Chen, and M. Lu, "A substation oriented approach to optimal phasor measurement units placement," *Journal of Electrical Engineering and Technology*, vol. 10, no. 1, pp. 18–29, Jan 2015.
- [12] T. Kerchuen and W. Ongsakul, "Optimal pmu placement by stochastic simulated annealing for power system state estimation," *GMSARN International Journal*, vol. 2, 2008.
- [13] R. F. Nuqui and A. G. Phadke, "Phasor measurement unit placement techniques for complete and incomplete observability," *IEEE Transactions on Power Delivery*, vol. 20, Oct 2005.
- [14] D. Gyllstrom, E. Rosensweig, and J. Kurose, "On the impact of pmu placement on observability and cross-validation," in *Proceedings of the 3rd International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet*, ser. e-Energy '12. New York, NY, USA: ACM, 2012.
- [15] J. Peng, Y. Sun, and H. Wang, "Optimal pmu placement for full network observability using tabu search algorithm," *International Journal of Electrical Power and Energy Systems*, vol. 28, 2006.
- [16] E. Abiri and F. Rashidi, "Optimal phasor measurement units placement to maintain network observability using a novel binary particle swarm optimization and fuzzy system," *J. Intell. Fuzzy Syst.*, vol. 28, Jan. 2015.
- [17] N. Xia, H. Gooi, S. Chen, and M. Wang, "Redundancy based pmu placement in state estimation," *Sustainable Energy, Grids and Networks*, vol. 2, pp. 23 – 31, 2015.
- [18] M. Carvalho, J. DeMott, R. Ford, and D. A. Wheeler, "Heartbleed 101," *IEEE Security Privacy*, vol. 12, no. 4, pp. 63–67, July 2014.
- [19] F. C. Schweppe and J. Wildes, "Power system static-state estimation, part i: Exact model," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-89, Jan 1970.
- [20] A. Mahari and H. Seyed, "Optimal pmu placement for power system observability using bica, considering measurement redundancy," *Electric Power Systems Research*, vol. 103, 2013.
- [21] X. Bei, Y. J. Yoon, and A. Abur, "Optimal placement and utilization of phasor measurements for state estimation," *PSERC Publication*, pp. 1550–1555, Aug. 2005.
- [22] A. Bretto, *Hypergraph Colorings*. Heidelberg: Springer International Publishing, 2013, pp. 43–56.
- [23] M. Campillo, R. Corra, and Y. Frota, "Cliques, holes and the vertex coloring polytope," *Information Processing Letters*, vol. 89, no. 4, 2004.
- [24] Michel Berkelaar. [Online]. Available: <http://web.mit.edu/lpsolve/doc/>
- [25] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.
- [26] M. Newman, "Western states power grid data set," *Retrieved on October 12, 2016*. [Online]. Available: <http://www-personal.umich.edu/~mejn/netdata/power.zip>



Teklemariam T. Tesfay obtained his B.Sc. degree from Mekelle Institute of Technology, Ethiopia in 2007, and his MSc degree from the Indian Institute of Technology, Bombay, India, in 2009. He received his Ph.D. degree from the École Polytechnique Fédérale de Lausanne (EPFL), Switzerland in 2017. He is currently a postdoctoral researcher at the LCA2 laboratory, EPFL, Switzerland. His research interests include identifying cybersecurity threats and proposing countermeasures for smart grid networks.



Jean-Yves Le Boudec is professor at EPFL and fellow of the IEEE. He graduated from Ecole Normale Supérieure de Saint-Cloud, Paris, where he obtained the Agrégation in Mathematics in 1980 and received his doctorate in 1984 from the University of Rennes, France. From 1984 to 1987 he was with INSA/IRISA, Rennes. In 1987 he joined Bell Northern Research, Ottawa, Canada, as a member of scientific staff in the Network and Product Traffic Design Department. In 1988, he joined the IBM Zurich Research Laboratory where he was manager

of the Customer Premises Network Department. In 1994 he became associate professor at EPFL. His interests are in the performance and architecture of communication systems and smart grids. He co-authored a book on network calculus, which forms a foundation to many traffic control concepts in the internet, an introductory textbook on Information Sciences, and is also the author of the book "Performance Evaluation".



Ola Svensson graduated from IDSIA in 2009; his advisor was Monaldo Mastrolilli. The subject of his thesis was the approximability of graph and scheduling problems. After spending two years as a postdoc with Johan Håstad at KTH Royal Institute of Technology, Sweden, he is now back in Switzerland as a faculty at the School of Computer and Communication Sciences, EPFL. His research interests include approximation algorithms, combinatorial optimization, and computational complexity. His work on the traveling salesman problem received the best

paper award at FOCS'11.